**EXERCICIOS**

1. Crie um programa que leia nome, sexo e idade de várias pessoas, guardando os dados de cada pessoa em um dicionário e todos os dicionários em uma lista. No final, mostre:
2. Quantas pessoas foram cadastradas
3. A média de idade
4. Uma lista com as mulheres
5. Uma lista de pessoas com idade acima da média

print()

print('\*\*\*\*\* SISTEMA DE CADASTRO \*\*\*\*\*')

lista = []

lista\_mulher = []

soma\_idade = 0

while True:

nome = input('Digite o seu nome: ').capitalize()

sexo = input('Digite seu sexo [M/F]?: ').upper()

idade = int(input('Digite sua idade: '))

dic = {'nome': nome, 'sexo': sexo, 'idade': idade}

lista.append(dic)

soma\_idade+= idade

if sexo == 'F':

lista\_mulher.append(sexo)

lista\_mulher.append(nome)

pergunta = input('Quer continuar com os cadastros [S/N]?: ').upper()

if not pergunta == 'S':

break

print()

print('\*\*\*\*\* ANÁLISE DO SISTEMA DE CADASTRO \*\*\*\*\*')

print('<<<>>>'\*10)

print(f'AO TODO FORAM REGISTRADAS {len(lista)} PESSOA/S')

print('<<<>>>'\*10)

print(f'A MÉDIA DAS IDADE/S É DE {soma\_idade/len(lista)} ANO/S DE IDADE')

print('<<<>>>'\*10)

print('LISTA DE MULHERES')

for lista\_mulher in lista:

if lista\_mulher['sexo'] == 'F':

print('Nome:', lista\_mulher['nome'], 'Idade:', lista\_mulher['idade'])

print('<<<>>>'\*10)

print('LISTA DE PESSOAS COM IDADE ACIMA DA MÉDIA')

for lista\_mulher in lista:

if(lista\_mulher['idade'] > (soma\_idade/len(lista))):

print('Nome:',lista\_mulher['nome'], 'Sexo:',lista\_mulher['sexo'], 'Idade:',lista\_mulher['idade'])

1. Escreva um programa Python para criar uma tupla com números de 1 a 4 e imprima um item

t1 = (1,2,3,4)

print(t1[2])

1. Escreva um programa Python para desempacotar uma tupla em várias variáveis:

considere uma tupla = (‘aluno’,’universidade’, ‘nota’, ‘resultado’) e as variáveis w, x, y, z

Consulte o tipo de dados de cada uma das variáveis e impriva os valores de cada uma delas.

t1 = ('aluno', 'universidade', 'nota', 'resultado')

w, x, y, z = t1

print(w, x, y, z)

1. Crie 3 conjuntos conforme estrutura a seguir:

setx = set(["apple", "mango"])

sety = set(["mango", "orange"]) setz = set(["mango"])

Faça as seguintes operações sobre conjuntos:

* 1. Faça a união dos três conjuntos e imprima o resultado
  2. Verifique quais os elementos comuns do conjunto setx e sety e imprima o resultado
  3. Verifique se o conjunto setx é subconjunto do conjunto sety e setz utilizando issubset()
  4. Verifique quais elementos do conjunto setx não existem em sety

print('\*\*\*\*\* a) \*\*\*\*\*')

setx = set(["apple", "mango"])

sety = set(["mango", "orange"])

setz = set(["mango"])

unidos = setx.union(sety, setz)

print(f'A união dos três conjunto:\n{unidos}')

print()

print('\*\*\*\*\* b) \*\*\*\*\*')

setx = set(["apple", "mango"])

sety = set(["mango", "orange"])

setz = set(["mango"])

setb = set(setx).intersection(sety)

print(f'Os elementos comuns do conjunto setx e sety:\n{setb}')

print()

print('\*\*\*\*\* c) \*\*\*\*\*')

setx = set(["apple", "mango"])

sety = set(["mango", "orange"])

setz = set(["mango"])

print(f'O conjunto setx é subconjunto sety: {setx.issubset(sety)}')

print(f'O conjunto setx é subconjunto setz: {setx.issubset(setz)}')

print()

print('\*\*\*\*\* d) \*\*\*\*\*')

setx = set(["apple", "mango"])

sety = set(["mango", "orange"])

setz = set(["mango"])

print(f'Quais elementos do conjunto setx não existem em sety:\n{setx.difference(sety)}')

print()

1. Escreva um programa Python para inserir um elemento no início de um determinado DicionárioOrdenado.

DicionárioOrdenado original:

DicionárioOrdenado ([('color1', 'Red'), ('color2', 'Green'), ('color3', 'Blue')]) Insira um elemento no início do referido DicionárioOrdenado: DicionárioOrdenado atualizado:

DicionárioOrdenado ([('color4', 'Orange'), ('color1', 'Red'), ('color2', 'Green'), ('color3', 'Blue')])

dicionario\_ordenado = {'color1' : 'Red',

'color2' : 'Green',

'color3' : 'Blue'}

dicionario\_ordenado.update({'color4' : 'Orange'})

for k, v in dicionario\_ordenado.items():

print(f'{k} = {v}')

1. Escreva uma função chamada right\_justify, que receba uma string chamada s como parâmetro e exiba a string com espaços suficientes à frente para que a última letra da string esteja na coluna 70 da tela:

>>> right\_justify('monty')

Dica: Use concatenação de strings e repetição. Além disso, o Python oferece uma função integrada chamada len, que apresenta o comprimento de uma string, então o valor de len('monty') é 5.

nome = str(input('Insira uma palavra: '))

def right\_justify(nome):

i=1

j=70-len(nome)

while i<=j:

nome = ''+nome

i += 1

return print(nome)

right\_justify(nome)

1. Um objeto de função é um valor que pode ser atribuído a uma variável ou passado como argumento. Por exemplo, do\_twice é uma função que toma um objeto de função como argumento e o chama duas vezes:

**def** do\_twice(f): f()

f()

Aqui está um exemplo que usa do\_twice para chamar uma função chamada print\_spam duas vezes:

**def** print\_spam(): print('spam')

do\_twice(print\_spam)

* 1. Digite este exemplo em um script e teste-o.
  2. Altere do\_twice para que receba dois argumentos, um objeto de função e um valor, e chame a função duas vezes, passando o valor como um argumento.

def do\_twice (func, arg):

"""Executa uma função duas vezes.

func: objeto de função

arg: argumento passado para a função

"""

func (arg)

func (arg)

def print\_twice (arg):

"""Imprime o argumento duas vezes.

arg: qualquer coisa imprimível

"""

print (arg)

print (arg)

def do\_four (func, arg):

"""Executa uma função quatro vezes.

func: objeto de função

arg: argumento passado para a função

"""

do\_twice (func, arg)

do\_twice (func, arg)

do\_twice (print, 'spam')

print ('')

do\_four (print, 'spam')

1. Crie um programa que faça a impressão de uma mensagem e a multiplicação de dois números. Utilize módulos e funções para resolução desse problema.
2. O usuário deve entrar com a mensagem e com o uso de módulos e funções, essa mensagem deve ser impressa na tela
3. O usuário deve entrar com os valores dos dois multiplicadores e o programa deve exibir o resultado na tela.
4. Dada uma lista encadeada de caracteres formada por uma sequência alternada de letras e dígitos, construa um método que retorne uma lista na qual as letras são mantidas na sequência original e os dígitos são colocados na ordem inversa. Exemplos:

A 1 E 5 T 7 W 8 G → A E T W G 8 7 5 1

3 C 9 H 4 Q 6 → C H Q 6 4 9 3

Como mostram os exemplos, as letras devem ser mostradas primeiro, seguidas dos dígitos. Sugestões:

* 1. usar uma fila e uma pilha;
  2. supor um método ehDigito() que retorna um valor booleano, como por exemplo, verdadeiro caso um caractere seja um dígito.

**Gostei de ver as implementações, você está indo muito bem!! Vi que na questão do print\_spam criou até uma função que imprime 4 vezes. Isso é ótimo para o consumo alimentar e está firme no conteúdo e mostra o quanto se dedica.**

**A, apresento a seguir:**

**5) O novo elemento ('color4', 'Orange') deve ficar na primeira posição do dicionário.**

**Para isso, grito o dicionário com a seguinte estrutura:**

**dic = ([('cor1', 'Vermelho'), ('cor2', 'Verde'), ('cor3', 'Azul')])**

**Depois, insira o novo elemento na posição 0 usando a função dic.insert()**

**dic.insert(0, [('color4', 'Laranja')])**

**Veja o resultado:**

**imprimir(dic)**

**6) Altere a linha nome = ''+nome**

**pára:**

**nome = ' '+ nome**

**é preciso colocar o espaço em branco entre aspas.**

**08) Não encontrei esta questão. Vale sobressair que um módulo é um arquivo .py que você vai criar e importar no código principal.**

**09) Não encontrei esta questão, segue dicas de como resolvê-la:**

**A função ehDigito deve ser fornecida para que seja dado um elemento a função de remuneração se ele é um dígito ou não.**

**Para que esta função seja capaz de informar se um elemento determinado é dígito ou não, seja necessário fazer o seguinte elemento:**

**Na definição da função um parâmetro. Este parâmetro será usado pela função. Em outras palavras, entre parâmetro é o elemento que será concluído (se ele é dígito)**

**def ehDigito( elemento ):**

**Feito isso, agora vamos implementar a ação desta função que é verificar se este elemento é um dígito. Para que a função identifique um dígito, primeiro ela precisa saber o que é isso. Então vamos criar uma lista contendo dígitos.**

**def ehDigito(elemento):**

**listaDigitos = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]**

**Para saber se a lista de elementos de parâmetros é um dígito ou seu valor precisa estar dentro dos dígitos corretos?**

**Então vamos fazer esta verificação:**

**def ehDigito(elemento):**

**listaDigitos = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]**

**se elemento na listaDigitos:**

**retornar Verdadeiro**

**Ou seja:**

**def ehDigito(elemento):**

**listaDigitos = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]**

**se o elemento estiver na listaDigitos :**

**Reembolso verdadeiro**

**Ok, nossa função já sabe informar se um elemento é dígito. Agora, sabe-se que se um elemento não é dígito, ele é uma letra. Então, como a função identifica se um número é dígito e, se ele for, a função verdadeira; quando o elemento não é um dígito ele precisa retornar falso, porque é uma letra. Para isso faça:**

**def ehDigito(elemento):**

**listaDigitos = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]**

**se elemento na listaDigitos:**

**retornar Verdadeiro**

**outro:**

**retorna falso**

**Ou seja:**

**def ehDigito(elemento):**

**listaDigitos = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]**

**se o elemento estiver na listaDigitos :**

**Reembolso verdadeiro**

**caso contrário:**

**retorno falso**

**Ótimo, nossa função está pronta! Agora precisamos usa-la. Vamos lembrar que temos, inicialmente, uma lista contendo dígitos e letras e precisamos organizar-la conforme o enunciado. Podemos usar a seguinte lógica:**

**1-primeiro você tem que percorrer a lista original (aquela que contém as letras e dígitos) usando para;**

**2-Dentro do for, você vai analisar cada elemento da lista é digito ou letra, para que você possa usar o método ehDigito que você criou, porque se o elemento não for um dígito, ele é uma letra;**

**A partir da seleção usando o método ehDigito que pode ser usado em listas diferentes de você. Ou seja, os elementos que antecedem os dígitos em uma lista e as letras em outra lista;**

**4-no final, mostre na tela as letras e, depois os primeiros dígitos na ordem inversa de inserção, ou seja, se a ordem original é: 1,5,3 deve ser balcão: 3,5,2**

**Então vamos lá:**

**1-primeiro você tem que percorrer a lista original (aquela que contém as letras e dígitos) usando para;**

**A lista é esta: ['A', 1 , 'E', 5, 'T', 7, 'W', 8, 'G']**

**Para isso, faça:**

**para elemento na lista:**

**Ou seja, a lista tem 9 elementos, então o for executar o código dentro de 9 vezes. Na primeira execução elemento='A', na segunda execução elemento=1, na terceira execução elemento='E' e assim sucessivamente…**

**2-Dentro do for, você vai analisar cada elemento da lista é digito ou letra, para que você possa usar o método ehDigito que você criou, porque se o elemento não for um dígito, ele é uma letra;**

**Para isso, faça:**

**para elemento na lista:**

**respostaDaFuncao=ehDigito(elemento)**

**Nós criamos a função ehDigimos para ela como parâmetro um elemento variável. Na primeira execução do for elemento='A', logo respostaDaFuncao= False porque A é letra e, não dígito. Na segunda execução do for, elemento=1, logo respostaDaFuncao= True porque 1 é dígito. Na terceira execução elemento='E', logo respostaDaFuncao= False porque E é letra e, não dígito. E assim sucessivamente…**

**Implemente e veja os resultados:**

**para elemento na lista:**

**respostaDaFuncao=ehDigito(elemento)**

**print(respostaDaFuncao)**

**A partir da seleção usando o método ehDigito que pode ser usado em listas diferentes de você. Ou seja, os elementos que antecedem os dígitos em uma lista e as letras em outra lista;**

**Vamos criar as listas:**

**dígitos=[]**

**letras=[]**

**Agora vamos completar as listas. Se o elemento para dígitos insira o seu valor na lista de dígitos , se não for, insira seu valor na lista de letras .**

**para elemento na lista:**

**respostaDaFuncao=ehDigito(elemento)**

**if respostaDaFuncao == Verdadeiro:**

**digitos.append(elemento)**

**outro:**

**letras.append(elemento)**

**Ou seja, percorremos todos os elementos da lista e verificamos com a função ehDigito se cada elemento era digital ou não. Quando o elemento era dígito, inserimos o seu valor na lista de dígitos ; quando não era dígito, inserimos seu valor na lista de letras .**

**Pronto, agora já sabemos quem é letra e quem é digitado dentro da nossa.**

**Implemente e veja:**

**print( letras)**

**imprimir ( digitos)**

**4-No final, mostre na tela as letras e, primeiro, os dígitos na ordem inversa de inserção depois, ou seja, se a ordem original é: 1,5,3; deve ser mostrado: 3,5,2**

**Na programação a gente consegue fazer operações entre strings, como por exemplo, adicionalmente. Veja:**

**”n” + ”n” = “nn”**

**”-” + “ \* ” + ”-” = “-\*-”**

**“oi” + ” “ = “oi”**

**Então, nós podemos fazer isso entre listas também, por exemplo:**

**['a' , 'b' , 'c'] + ['d' , 'e'] = ['a' , 'b' , 'c' , 'd' , 'e']**

**Como no enunciado pede para que, primeiro, mostremos as letras e, depois, os dígitos na ordem inversa da inserção, vamos fazer:**

**letras + digitos[::-1]**

**Implemente e veja o resultado:**

**print(letras + digitos[::-1])**

**[::-1] -> uma lista ou string ao contrário (mostra na inversa da inserção).**

**---**

**Qualquer dúvida, chame em nosso grupo do Discord no módulo referente :)**